IN THE CLAIMS:

1. (Previously Presented) An SMBus host controller comprising:

an SMBus interface; and

an SMBus message handler including:

a memory storing microcode comprising at least two programs each for handling a bus command protocol, each program comprising at least one instruction;

an interface to a register configured to identify a starting address of a program in said memory;

an instruction fetch unit configured to read an instruction at an address in said memory, said address being specified by a program counter;

a finite-state machine configured to receive and interpret the instructions read by said instruction fetch unit and manage the data transfer between the SMBus interface and a register set in compliance with said instructions read from said memory; and

an address register array comprising a plurality of starting addresses of programs stored in said memory, said register comprising an offset for pointing at a specific register in said address register array.

2. (Previously Presented) The SMBus host controller of claim 1, wherein said register set complies with the ACPI specification.

3. (Cancelled).

4. (Previously Presented) The SMBus host controller of claim 2, further comprising a buffer pointer register for pointing at one of a plurality of data registers said

finite-state machine transferring data read from the SMBus interface to the data register at which said buffer pointer register points if said finite-state machine interprets a reception instruction; said finite-state machine transferring the data read from the data register at which said buffer pointer register points to said SMBus interface if said finite-state machine interprets a transmission instruction.

5. (Previously Presented) The SMBus host controller of claim 4, wherein the finite-state machine causes said buffer pointer register to be incremented each time a reception instruction or a transmission instruction is executed.

6. (Previously Presented) The SMBus host controller of claim 1, further comprising a loop counter for storing the value of a block counter register in said loop counter if the finite-state machine executes an instruction to transmit data from said block counter register to said SMBus interface, said loop counter being decremented each time a data byte is transmitted to said SMBus interface while an instruction to transmit data from said block counter register to said SMBus interface is executed, wherein the instruction to transmit data from said block counter register to said SMBus interface is completed when the value of said loop counter reaches zero.

7. (Previously Presented) The SMBus host controller of claim 1, further comprising a loop counter and a block count register both for storing a byte received from said SMBus interface if the finite-state machine executes an instruction to receive data at said block counter register from said SMBus interface, said loop counter being decremented each time a data byte is transmitted to or received from said SMBus interface while an instruction to receive data at said block counter register from said SMBus interface is executed, wherein the instruction to receive data at said block counter register from said SMBus interface is completed when the value of said loop counter reaches zero.

8. (Previously Presented) The SMBus host controller of claim 1, wherein each instruction comprises one bit indicating as to whether or not an instruction is the last instruction in the program.

9. (Previously Presented) The SMBus host controller of claim 1, wherein each instruction comprises one bit indicating as to whether an instruction is to be executed only once or this instruction is to be executed repeatedly until a loop counter becomes zero, wherein said loop counter is decremented each time an instruction is executed repeatedly.

10. (Previously Presented) An integrated circuit chip for transmitting and receiving data over an SMBus, the integrated circuit chip comprising:

an SMBus interface;

an interface to a memory storing microcode comprising at least two programs each for handling a bus command protocol, each program comprising at least one instruction;

an interface to a register configured to identify a starting address of a program in said memory;

an instruction fetch unit configured to read an instruction at an address in said memory; said address being specified by a program counter (pc);

a finite-state machine configured to receive and interpret the instructions read by said instruction fetch unit and to manage the data transfer between the SMBus interface, and a register set in compliance with said instructions read from said memory;

an address register array comprising a plurality of starting addresses of programs stored in said memory, said register comprising an offset for pointing at a specific register in said address register array.

11. (Original) The integrated circuit chip of claim 10, wherein said register set complies with the ACPI specification.

12. (Cancelled).

13. (Previously Presented) The integrated circuit chip of claim 11, further comprising a buffer pointer register for pointing at one of a plurality of data registers comprised in register set; said finite-state machine transferring data read from the SMBus interface to the data register at which said buffer pointer register points if said finite-state machine interprets a data reception instruction, said finite-state machine transferring the data read from the data register at which said buffer pointer register points to said SMBus interface if said finite-state machine interprets a data transmission instruction.

14. (Previously Presented) The integrated circuit chip of claim 13, wherein the finite-state machine causes said buffer pointer register to be incremented each time a data reception instruction or a data transmission instruction is executed.

15. (Previously Presented) The integrated circuit chip of claim 10, further comprising a loop counter for storing the value of a block counter register SMB_BCNT in said loop counter if the finite-state machine executes an instruction to transmit data from said block counter register to said SMBus interface, said loop counter being decremented each time a data byte is transmitted to said SMBus interface while an instruction to transmit data from said block counter register to said SMBus interface is executed, wherein the instruction to transmit data from said block counter register to said SMBus interface is completed when the value said loop counter reaches zero.

16. (Previously Presented) The integrated circuit chip of claim 10, further comprising a loop counter and a block count register comprised in said register set both for storing a byte received from said SMBus interface if the finite-state machine executes an instruction to receive data at said block counter register from said SMBus interface, said loop counter being decremented each time a data byte is transmitted to or received from said SMBus interface while an instruction to a receive data at said block counter register is executed, wherein the instruction to receive data at said block counter register from said SMBus interface is completed when the value of said loop counter reaches zero.

17. (Previously Presented) The integrated circuit chip of claim 10, wherein each instruction comprises one bit indicating as to whether or not an instruction is the last instruction in the program.

18. (Previously Presented) The integrated circuit chip of claim 10, wherein each instruction comprises one bit indicating as to whether an instruction is to be executed only once or this instruction is to be executed repeatedly until a loop counter becomes zero, wherein said loop counter is decremented each time an instruction is executed repeatedly.

19. (Previously Presented) A method for controlling an SMBus, the method comprising:

storing, in a memory, microcode comprising at least two programs each for handling a bus command protocol, each program comprising at least one instruction;

identifying a starting address of one of the at least two programs stored in the memory;

fetching instructions of said program one after another and receiving the instructions into a finite-state machine;

interpreting the instructions;

transferring data between an SMBus interface and a register set in compliance with the instruction present in said finite-state machine;

reading a first value of a protocol register specifying an offset in an address register array; and

reading a second value of a register of said address register array, said register being specified by said offset; said second value constituting said starting address of said program.

20. (Previously Presented) The method of claim 19, wherein said register set complies with the ACPI specification.

21. (Cancelled).

22. (Previously Presented) The method of claim 20, wherein said transferring step comprising the sub-steps of:

interpreting a reception instruction;

reading the value of a buffer pointer register; and

transferring the data read from said SMBus interface to the data register at which the value stored in said buffer pointer register points.

23. (Previously Presented) The method of claim 22, wherein said transferring step further comprises incrementing said value of said buffer pointer register.

24. (Previously Presented) The method of claim 22, wherein said transferring step further comprising decrementing a loop counter and checking as to whether said loop counter has a value of zero.

25. (Previously Presented) The method of claim 20, wherein said transferring step comprises the sub-steps of:

interpreting a transmission instruction;

reading the value of a buffer pointer register; and

transferring the data read from the data register at which the value stored in said buffer pointer register to said SMBus interface.

26. (Previously Presented) The method of claim 25, wherein said transferring step further comprises incrementing of said buffer pointer register.

27. (Previously Presented) The method of claim 25, wherein the transferring step further comprises decrementing of said loop counter.

28. (Previously Presented) The method of claim 19, wherein said transferring step comprises:

Interpreting an instruction to transmit data from a block counter register to said SMBus interface;

storing the value of said block count register in a loop counter; and

transmitting the value of said block counter register to said SMBus interface.

29. (Previously Presented) The method of claim 19, wherein said transferring step comprises:

interpreting an instruction to receive data from said SMBus interface;

transmitting a byte from said SMBus interface to said block count register; and

storing the value of the byte received from said SMBus interface to a loop counter register.

30. (Previously Presented) The method of claim 19, wherein said transferring further comprises:

determining as to whether a stop bit has a predetermined value; if this is the case:

writing 80h into a status register of said register set.

31. (Previously Presented) The method of claim 19, wherein said transferring further comprises:

determining as to whether a loop bit of an instruction has a predetermined value; if that is the case,

executing said instruction repeatedly;

decrementing a loop counter each time said instruction is executed;

finishing the execution of said loop instruction when the value of the said loop counter becomes zero; and

fetching the next instruction.

32-95. (Cancelled).

96. (Previously Presented) The SMBus host controller of claim 1, wherein the memory storing microcode is a read-only memory.